Using NHibernate mapping to perform CRUD operations

1. Create a new WPF Project / In the existing Project

2. Install
     a) Nhibernate using NuGet(Soultion-> Project->References -> Manage NuGet Packages)

     b) add refernce to System.Configuration

3. Create a .cs Class with the required Properties.
     e.g

```
     namespace FSU.Application
     {
        public class Person
       {
      public virtual string FirstName { get; set; }
        public virtual string LastName { get; set; }
        public virtual Using string MiddleName { get; set; }
        }
      }
```

 Note: virtual keyword is required to support Lazy loading

4. NHibernate is used to persit instances of an entity in a (relational) database.An instance of an entity in the domain corresponds to a row in a table in the database. So, define a mapping between the entity and the corresponding table in the database. This mapping can be done either by defining a mapping file (an xml-document) or by decorating the entity with attributes.

5. Create a new xml document in the project and call it Person.hbm.xml. The "hbm" part of the file name, is a convention used by NHibernate to automatically recognize the file as a mapping file. Define "Embedded Resource" as Build Action for this xml file.

6. In the Properties tab of Person.hbm.xml file
   a) Click the browse button on Schemas
   b) click Add button
   c) Browse to locate the nhibernate-mapping.xsd in the src folder of NHibernate. Click open, click OK.
   This xml schema definition file will be used when defining the mapping files. Visual Studio will then provide intellisense and validation when editing an xml mapping document.

7. Each mapping file has to define a <hibernate-mapping> root node.

```
   <?xml version="1.0" encoding="utf-8" ?>
  <hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
                   assembly="FSU.Application"
                   namespace="FSU.Application">

  <!-- more mapping info here -->

  </hibernate-mapping>
```

8. Define a primary key for the Product entity. You can use any property of the Person entity that would be unique or add a property called UserId and set its data type to Guid. The data type can

also be int or long etc.

  e.g

```
  using System;
namespace FSU.Application {
public class Person {
public virtual  Guid UserId { get; set; }
public virtual string FirstName { get; set; }
public virtual string LastName { get; set; }
public virtual string MiddleName { get; set; }
   }
}
```

 9. The complete mapping file

```
   <?xml version="1.0" encoding="utf-8" ?>
   <hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
                   assembly="FSU.Application"
                   namespace="FSU.Application">

    <class name="Person">
   <id name="UserID" column="UserID" >
      <generator class="assigned" /> // since the user is specifing the ID's the
generator is set to assigned
    </id>

    <property name="FirstName" column="FirstName" type="String" length="50"/>
    <property name="LastName" column="LastName" type="String" length="50"/>
    <property name="MiddleName" column="MiddleName" type="String"  length="50" />
   </class>
  </hibernate-mapping>
```

10. If you don't explicitly provide a column name for a property, NHibernate will name the column according to the property.
    NHibernate can automatically infer the name of the table or the type of the column from the class definition.

11. The database to be used with NHibernate is speicified in the connection details in the form of a connection string
    Add a new xml file to the FSU.Application project and call it App.config. Set its property "Copy to Output" to "Copy always".
    Set the Configuration file as follows

```
<configuration>

  <connectionStrings>
    <add name="FSUConnection" connectionString="Data
Source=(local)\SQLExpress;Initial Catalog=RFID;Integrated
Security=True;Pooling=False"
     providerName="System.Data.SqlClient" />
  </connectionStrings>

    <hibernate-configuration xmlns="urn:nhibernate-configuration-2.2">
        <session-factory>
            <property name="connection.provider">
                NHibernate.Connection.DriverConnectionProvider
            </property>
            <property name="dialect">
                NHibernate.Dialect.MsSql2008Dialect
            </property>
            <property name="connection.driver_class">
                NHibernate.Driver.SqlClientDriver
            </property>
            <property name="connection.connection_string_name">
            FSUConnection
          </property>
          <property name="show_sql">True</property>
        </session-factory>
    </hibernate-configuration>

</configuration>
```

This configuration tells NHibernate to use MS SQL Sever 2008 Edition as the target database and the name of the database shall be RFID.
By setting the show_sql to true one can see the SQL, Nhibernate generates and sends to the database.

12. Use SQL Server Management Studio to create the RFID Database.

13. Create a class called PersistenceManager.cs and set the configuration as follows

```
      public class PersistenceManager  {

        private ISessionFactory m_SessionFactory = null;
        private ISession m_Session = null;
        private string con =
ConfigurationManager.ConnectionStrings["FSUConnection"].ConnectionString;

        public PersistenceManager() {
            this.ConfigureNHibernate();
        }

        // <summary>
        // Configures NHibernate and creates a member-level session factory.
       // </summary>
        private void ConfigureNHibernate() {
            // Initialize
            NHibernate.Cfg.Configuration cfg = new NHibernate.Cfg.Configuration();
            cfg.Configure();

            /* Note: The AddAssembly() method requires that mappings be
             * contained in hbm.xml files whose BuildAction properties
             * are set to 'Embedded Resource'. */

            // Add class mappings to configuration object
            Assembly thisAssembly = typeof(Person).Assembly;
            cfg.AddAssembly(thisAssembly);

            // Create session factory from configuration object
            m_SessionFactory = cfg.BuildSessionFactory();
        }

    }
```

14. Use the m_session and m_sessionFactory to perofrm the CRUD operation on Database.